Isar — a Generic Interpretative Approach to Readable Formal Proof Documents

Markus Wenzel
TU Munich

September 1999

Overview

1. Notions of proof
2. Isar in a nutshell
3. Isar proof language
    - Syntax
    - Semantics
    - Derived concepts
4. Isabelle/Isar

Motivation

State of the art:

- many successful applications of semi-automated theorem proving systems
- output cryptic, inaccessible to non-experts

$\approx$ macro assembly language

Assumption: primary interest in proofs *per se* (communication, maintenance)

Goal: formal proof document preparation system, integrating

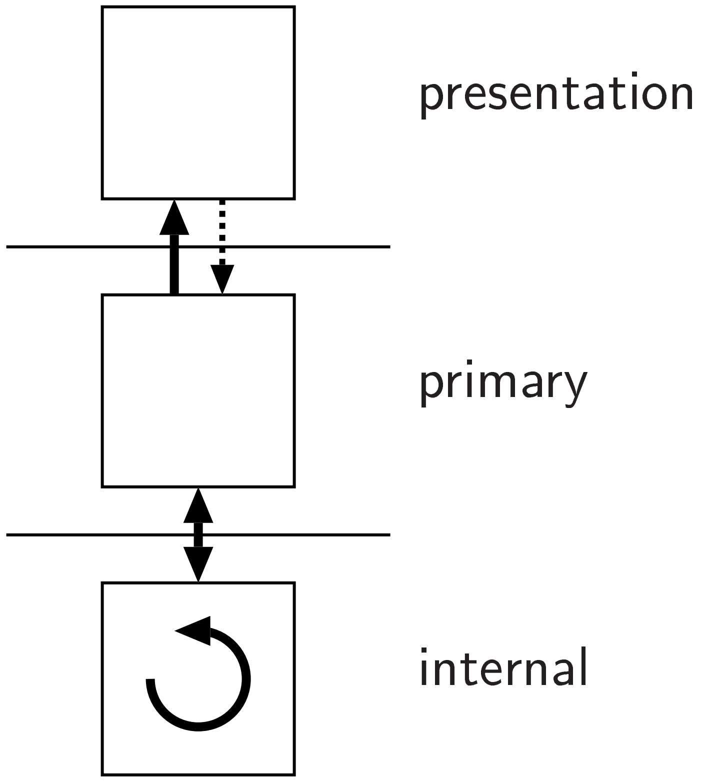
- machine proof checking
- readable representation

$\rightarrow$ Intelligible semi-automated reasoning

Notions of proof

Consider proofs at 3 levels:

internal: underlying inference engine
primary: view of author (input / output)
presentation: view of audience (output / feedback)

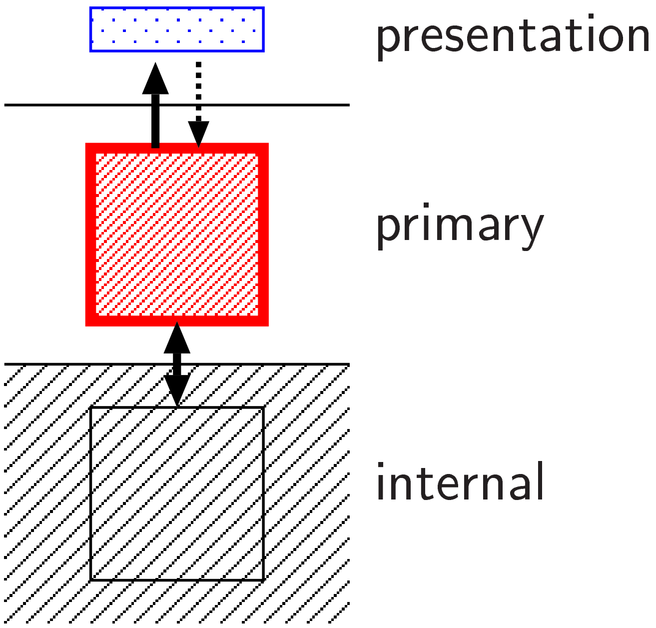


Claim: most systems do not have an appropriate primary notion of proof

Isar in a nutshell

Notions of proof in Isar:

- internal: take "truth" judgement as parameter, employ simple logical framework
- primary: the Isar proof language
- presentation: thin layer sufficient, Isar language designed with direct presentation in mind



Proof language desiderata:

1. intelligible
2. suitable for machine checking, including support for automation

Example Isar proofs (1)

lemma $A \longrightarrow A$
proof
  assume A
  show A **by** $(\mathit{assumption})$
qed

lemma $A \longrightarrow B \longrightarrow A$
proof
  assume A
  show $B \longrightarrow A$
  proof
    show A **by** $(\mathit{assumption})$
  qed
qed

lemma $(\exists x.\ P(f(x))) \longrightarrow (\exists x.\ P(x))$
proof
  assume $\exists x.\ P(f(x))$
  then show $\exists x.\ P(x)$
  proof $(\mathit{rule\ ex\text{-}elim})$ — — rule: $\dfrac{\exists x.\ A(x) \qquad \begin{array}{c}[A(x)]_x \\ \vdots \\ B\end{array}}{B}$
    fix a
    assume $P(f(a))$ (**is** $P(\mathit{??witness})$)
    show $\mathit{??thesis}$ **by** $(\mathit{rule\ ex\text{-}intro}\ [\mathit{of}\ P\ \mathit{??witness}])$
  qed
qed

Isabelle tactic scripts

```
Goal "A --> A";
  by (rtac impI 1);
  by (assume_tac 1);
qed "it";

Goal "A --> B --> A";
  by (rtac impI 1);
  by (rtac impI 1);
  by (assume_tac 1);
qed "it";

Goal "(EX x. P(f(x))) --> (EX x. P(x))";
  by (rtac impI 1);
  by (etac exE 1);
  by (rtac exI 1);
  by (assume_tac 1);
qed "it";
```

Main problems:

- script provides control information only
- replay of proof state history required

Isar basic proof language

Design goals:

- lean, generic, extensible
- support interactive work
- accommodate both primitive inferences and automation
- do not require proof search for basic operation

Main language elements:

fix x	context $\forall x.\ \square$
assume a: φ	context $\varphi \Longrightarrow \square$
have a: φ	intermediate goal
show a: φ	establish goal
note $a = a_1 \ \ldots \ a_n$	reconsider results
then	forward chaining
proof (m_1) ... **qed** (m_2)	proof methods

Language semantics:

1. declarative: abstract natural deduction proofs
2. operational: instructions of Isar virtual machine

Isar syntax

Primitives:

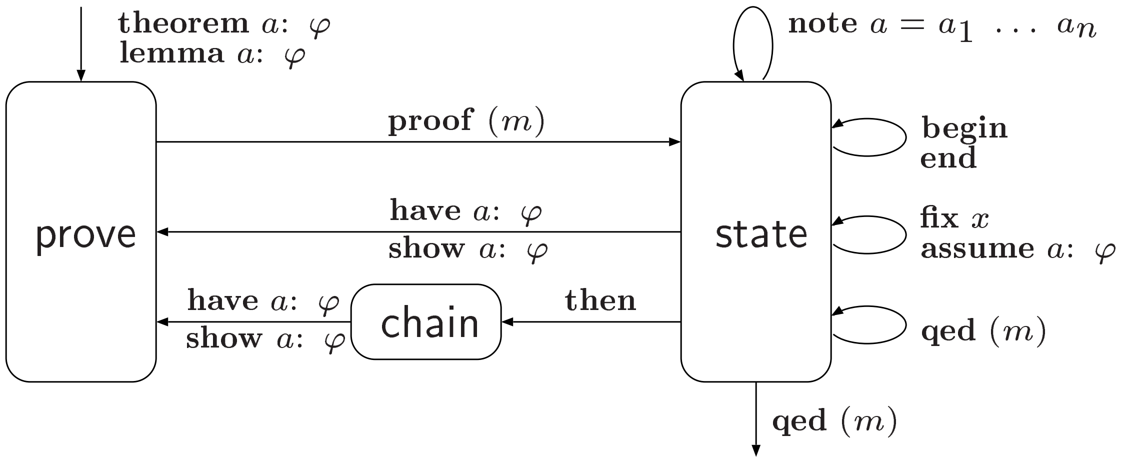
$$
\begin{array}{rcl}
\textit{theory-stmt} & = & \textbf{theorem } [\textit{name}\!:] \ \textit{form proof} \\
 & | & \textbf{lemma } [\textit{name}\!:] \ \textit{form proof} \\
 & | & \textbf{types } \ldots \ | \ \textbf{consts } \ldots \\
 & | & \textbf{defs } \ldots \ | \ \ldots \\
\textit{proof} & = & \textbf{proof } [(\textit{method})] \ \textit{stmt}^* \ \textbf{qed } [(\textit{method})] \\
\textit{stmt} & = & \textbf{begin } \textit{stmt}^* \ \textbf{end} \\
 & | & \textbf{note } \textit{name} = \textit{name}^+ \\
 & | & \textbf{fix } \textit{var}^+ \\
 & | & \textbf{assume } [\textit{name}\!:] \ \textit{form}^+ \\
 & | & \textbf{then } \textit{goal-stmt} \\
 & | & \textit{goal-stmt} \\
\textit{goal-stmt} & = & \textbf{have } [\textit{name}\!:] \ \textit{form proof} \\
 & | & \textbf{show } [\textit{name}\!:] \ \textit{form proof}
\end{array}
$$

Abbreviations:

$$
\begin{array}{rcl}
\textbf{from } a_1 \ \ldots \ a_n & \equiv & \textbf{note } \mathsf{this} = a_1 \ \ldots \ a_n \ \textbf{then} \\
\textbf{with } a_1 \ \ldots \ a_n & \equiv & \textbf{from } a_1 \ \ldots \ a_n \ \mathsf{this} \\
\textbf{hence} & \equiv & \textbf{then have} \\
\textbf{thus} & \equiv & \textbf{then show} \\
\textbf{by } (m_1) \ (m_2) & \equiv & \textbf{proof } (m_1) \ \textbf{qed } (m_2) \\
.. & \equiv & \textbf{by } (\textit{rule}) \\
. & \equiv & \textbf{by } (\textit{assumption})
\end{array}
$$

Isar operational semantics

Isar virtual machine:



Machine configuration: (block structured)

$$
\begin{array}{rcl}
\mathit{mode} & : & \textsf{prove} \mid \textsf{state} \mid \textsf{chain} \\
\mathit{fixes} & : & \mathit{list\ of\ var} \\
\mathit{asms} & : & \mathit{list\ of\ form} \\
\mathit{results} & : & \mathit{name} \rightharpoonup \mathit{list\ of\ theorem} \\
\mathit{problem} & : & (\mathit{bool} \times \mathit{name}) \times \\
 & & \quad ((\mathit{list\ of\ theorem}) \times \mathit{theorem})
\end{array}
$$

Transitions: $\mathcal{T}[\![I]\!] : \mathit{config} \rightharpoonup \mathit{config}$ for any Isar/VM instruction $I \in \{\mathbf{fix}\ x, \mathbf{assume}\ a\colon\ \varphi, \ldots\}$

Goal refinement

Proof script:

goal φ
  apply (m_1)
  apply (m_2)
  $\vdots$
  apply (m_k)
qed

$\longrightarrow$ Arbitrary tactic application considered harmful!

Proof document (backward-only view):

show φ
proof (m_1)
  $\vdots$
  fix $\vec{x}_i$ **assume** Γ_i
  show ψ_i $\ldots$
  $\vdots$
  fix $\vec{x}_j$ **assume** Γ_j
  show ψ_j $\ldots$
  $\vdots$
qed (m_2)

Extra-logical concepts

Term bindings:

- environment of higher-order text variables (subject to block structure)
- binding via higher-order unification
- automatic binding of "$?\!?thesis$" and "$\ldots$"

Example (1): explicit **let** statement

$\mathbf{let}\ ?\!?lhs < x \;=\; ?\!?thesis$

Example (2): impromptu views on propositions

$\mathbf{show}\ \varphi[n]\ (\mathbf{is}\ ?\!?P(n))$
$\mathbf{proof}\ (induct\ n)$
$\quad\mathbf{show}\ ?\!?P(0)\ \langle proof\rangle$
$\quad\mathbf{fix}\ \ n$
$\quad\mathbf{assume}\ ?\!?P(n)$
$\quad\mathbf{show}\ ?\!?P(Suc(n))\ \langle proof\rangle$
$\mathbf{qed}$

$\rightarrow$ abstract proofs

Example (3): calculations with "$\ldots$"

Derived concepts

Calculational proof: implicit reasoning with transitivity of $=$, $\leq$, $<$ etc.

Basic example (group theory):

$$
\begin{array}{l}
\textbf{lemma}\ \textit{right-unit}\colon\ x \circ 1 = x \\
\textbf{proof}\ (-) \\
\quad \textbf{have}\ x \circ 1 = x \circ (x^{-1} \circ x) \\
\quad\quad \textbf{by}\ (\textit{simp left-inverse}) \\
\quad \textbf{also have}\ \ldots = (x \circ x^{-1}) \circ x \\
\quad\quad \textbf{by}\ (\textit{simp assoc}) \\
\quad \textbf{also have}\ \ldots = 1 \circ x \\
\quad\quad \textbf{by}\ (\textit{simp right-inverse}) \\
\quad \textbf{also have}\ \ldots = x \\
\quad\quad \textbf{by}\ (\textit{simp left-unit}) \\
\quad \textbf{finally show}\ ?\textit{?thesis}\ . \\
\textbf{qed}
\end{array}
$$

Isar calculational elements:

$$
\begin{array}{rcl}
\textbf{also}_0 & \equiv & \textbf{note}\ \mathsf{calculation} = \mathsf{this} \\
\textbf{also}_n & \equiv & \textbf{note}\ \mathsf{calculation} = \textit{trans}\ \mathsf{calculation}\ \mathsf{this} \\
\textbf{finally} & \equiv & \textbf{also}\ \textbf{from}\ \mathsf{calculation}
\end{array}
$$

Example Isar proofs (2)

Advanced calculations:

lemma $\sum_{i<n} 2 \cdot i + 1 = n^2$ (**is** $?\!?P(n)$ **is** $?\!?S(n) = \_$)
proof $(induct\ n)$
  show $?\!?P(0)$ **by** $(simp)$

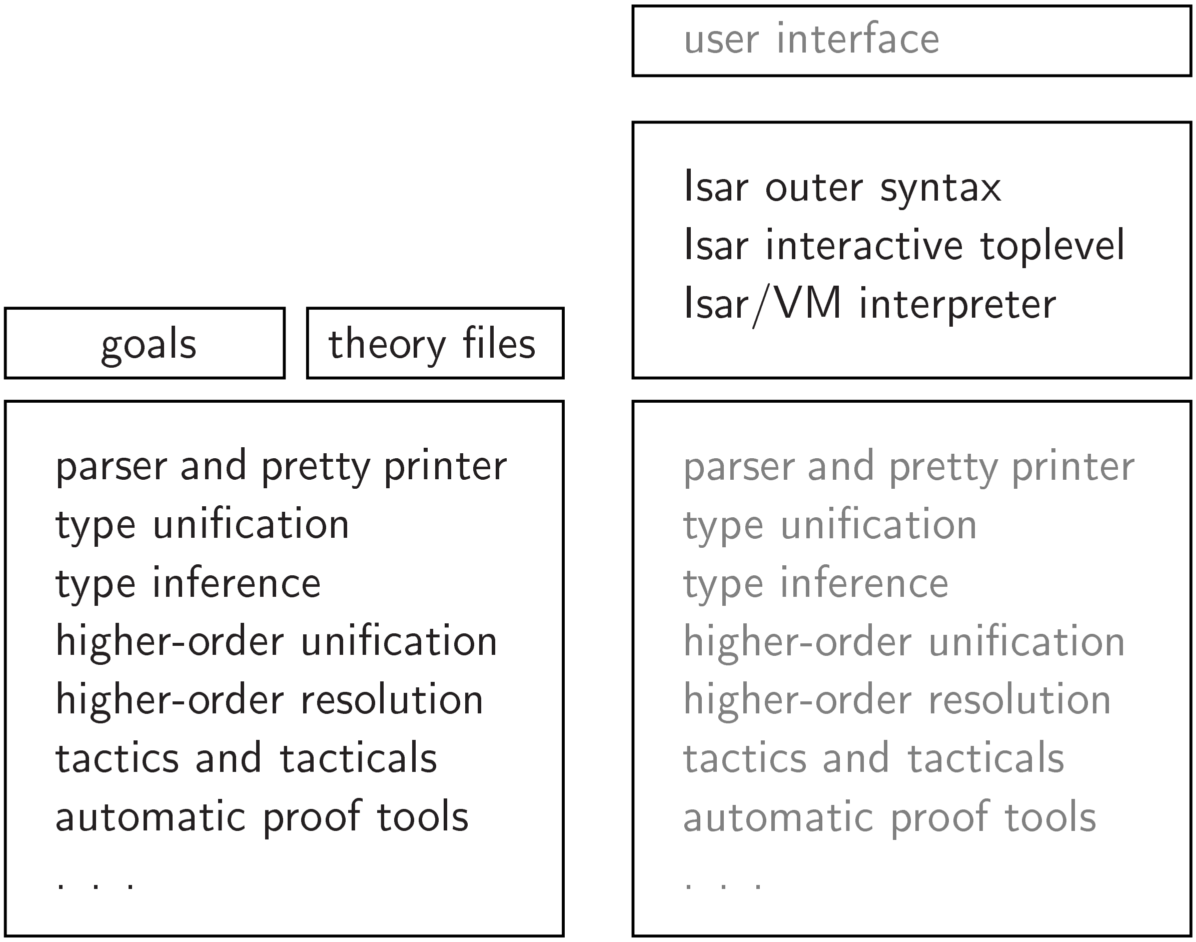
  fix n
  have $?\!?S(n+1) = ?\!?S(n) + 2 \cdot n + 1$
    by $(simp)$
  also assume $?\!?S(n) = n^2$
  also have $\ldots + 2 \cdot n + 1 = (n+1)^2$
    by $(simp)$
  finally show $?\!?P(Suc(n))$
    by $(simp)$
qed

Note:

- casual reasoning with substitution rules
- natural deduction and calculational proofs reconciled

Isabelle/Isar

Isabelle/Isar: real working environment for Isar proof document preparation, part of Isabelle99



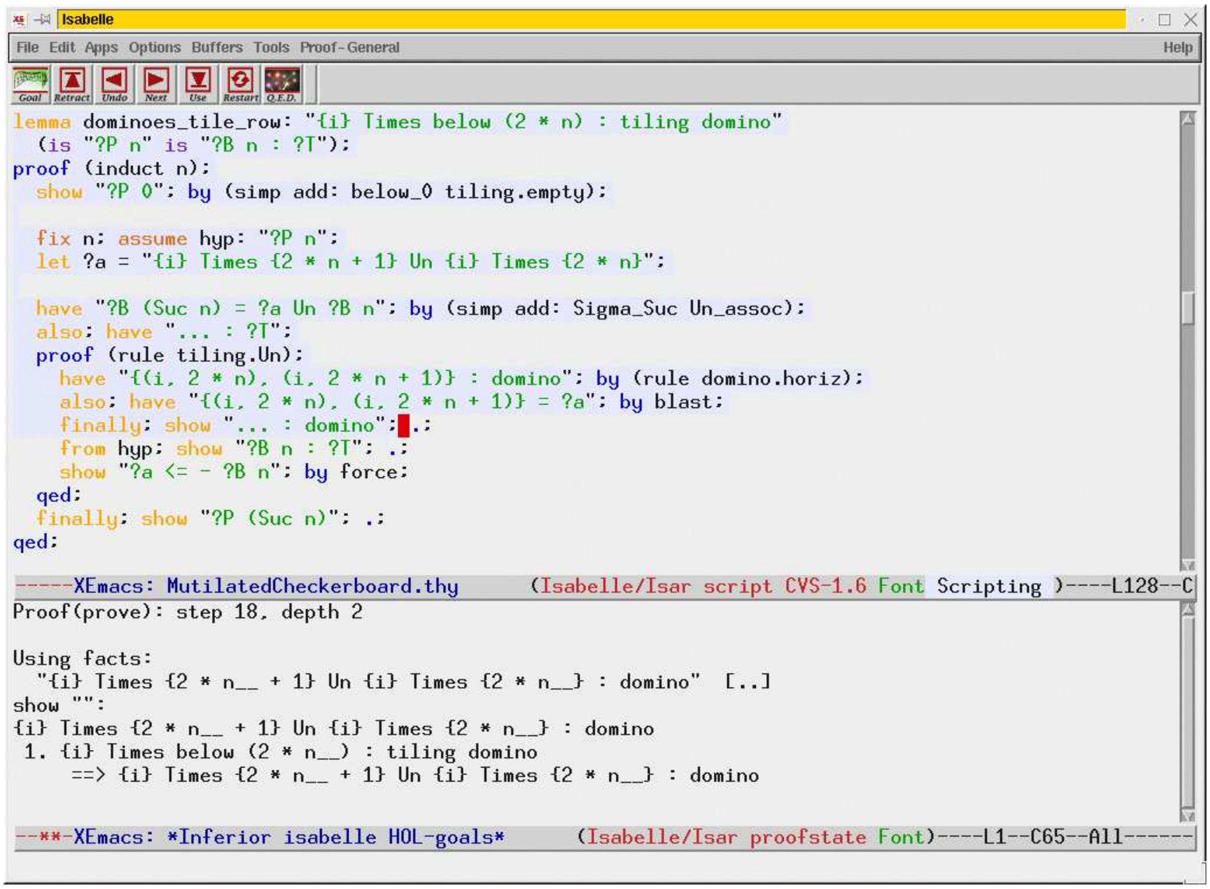
Isabelle classic

Isabelle/Isar

User interface

Proof General (LFCS Edinburgh): generic Emacs-based user interface for interactive theorem proving systems

→ achieve live Isar proof document editing



Applications

Examples:

- `KnasterTarski` ($\approx$ 0.5 pages): textbook-style reasoning, mixing forward / backward natural deduction and calculations
- `MutilatedCheckerboard` ($\approx$ 4 pages): reasoning with inductive sets
- `MultisetOrder` ($\approx$ 2 pages): tricky inductions
- `HahnBanach` ($\approx$ 20 + 20 pages): big mathematics application (by Gertrud Bauer)

Web page: `http://isabelle.in.tum.de/Isar/`